BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-074253
(43)Date of publication of application : 15.03.2002

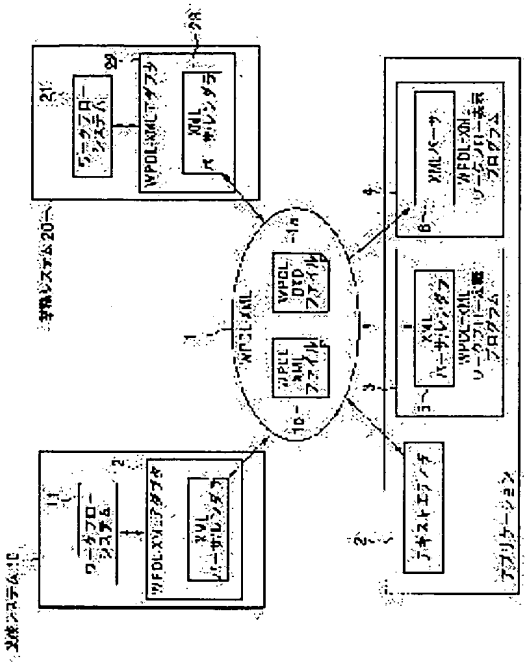
(51)Int.Cl. G06F 19/00
G06F 3/00
G06F 17/60

(21)Application number : 2000-267676
(22)Date of filing : 04.09.2000

(71)Applicant : TOSHIBA CORP
TOSHIBA SYSTEMS DEVELOPMENT CO LTD
(72)Inventor : TERASHITA YOSUKE
YOSHIMOTO TAKEHIRO

(54) METHODS FOR DEFINING, DISPLAYING, REUSING AND CONVERTING WORK FLOW

(57)Abstract:
PROBLEM TO BE SOLVED: To make easily performable cooperation between different kinds of work flow systems.
SOLUTION: This work flow defining method is characterized by providing a step to display icon parts for an operation concerning work flow definition on a screen, a step to define the work flow by arranging the icon parts on the screen, a step to generate a text character string for describing work flow definition in XML on the basis of document type definition which is stipulated for describing work flow definition in XML and a step to output the generated text character string.



LEGAL STATUS

[Date of request for examination] 21.02.2003
[Date of sending the examiner's decision of rejection]
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2002－74253
（P2002－74253A）
(43)公開日　平成14年３月15日(2002.3.15)

(51)Int.Cl.'		識別記号	Ｆ Ｉ		テーマコード（参考）
Ｇ０６Ｆ	19/00	３００	Ｇ０６Ｆ 19/00	３００Ｎ	５Ｂ０４９
	3/00	６５１	3/00	６５１Ａ	５Ｅ５０１
	17/60	１６２	17/60	１６２Ｃ	

審査請求　未請求　請求項の数６　ＯＬ　（全 20 頁）

(21)出願番号　特願2000－267676(P2000－267676)

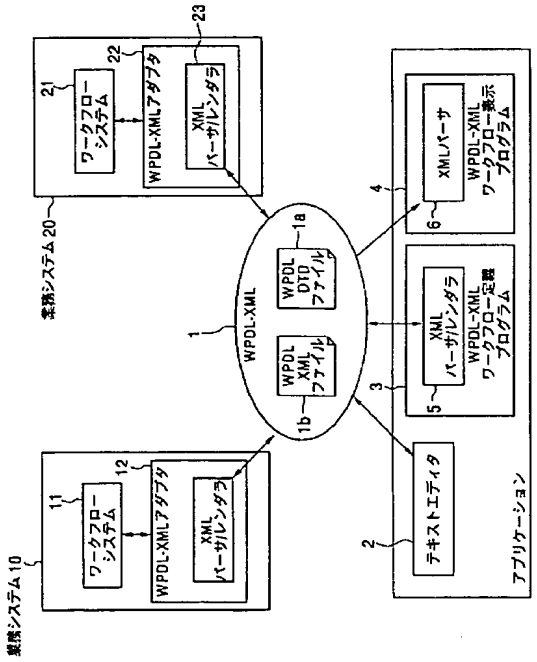
(22)出願日　平成12年９月４日(2000.9.4)

(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(71)出願人　000221100
東芝システム開発株式会社
東京都港区芝浦１丁目１番１号　東芝ビルディング
(72)発明者　寺下　陽介
東京都港区芝浦１丁目１番１号　株式会社東芝本社事務所内
(74)代理人　100077849
弁理士　須山　佐一

最終頁に続く

(54)【発明の名称】　ワークフロー定義方法、ワークフロー表示方法、ワークフロー再利用方法、ワークフロー変換方法

(57)【要約】
【課題】　異種ワークフローシステム間の連携を容易に行えるようにする。
【解決手段】　このワークフロー定義方法は、ワークフロー定義に関する操作用のアイコン部品を画面に表示する段階と、アイコン部品が画面上に配置されることでワークフローを定義する段階と、ワークフローの定義をXMLで表記するために規定した文書型定義に基づき、ワークフローの定義をXMLで表記したテキスト文字列を生成する段階と、生成されたテキスト文字列を出力する段階とを有することを特徴とする。



1

【特許請求の範囲】

【請求項１】　ワークフロー定義に関る操作用のアイコン部品を画面に表示する段階と、
前記アイコン部品が前記画面上に配置されることでワークフローを定義する段階と、
ワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Languageで表記したテキスト文字列として出力する段階とを有することを特徴とするワークフロー定義方法。

【請求項２】　ワークフロー処理が実行されているシステムより、ワークフローの進捗状況の情報をExtensible Markup Language で表記したテキスト文字列で取得する段階と、
取得されたテキスト文字列を、ワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの形態で表示する段階とを有することを特徴とするワークフロー表示方法。

【請求項３】　あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列を解析する段階と、
解析されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成する段階と、
生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成する段階と、
生成されたワークフローモデルを該当ワークフローシステムに登録する段階とを有することを特徴とするワークフロー定義変換方法。

【請求項４】　あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換する段階と、
変換されたテキスト文字列を他のワークフローシステムへ伝送する段階と、
伝送されてきたテキスト文字列を、自身のワークフローに変換する段階とを有することを特徴とするワークフロー再利用方法。

【請求項５】　ワークフロー定義に関る操作用のアイコン部品を画面に表示する段階と、
前記アイコン部品が前記画面上に配置されることでワークフローを定義する段階と、
ワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Languageで表記したテキスト文字列に変換する段階と、
変換されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成する段階と、

2

生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成する段階と、
生成されたワークフローモデルを該当ワークフローシステムに登録する段階とを具備したことを特徴とするワークフロー変換方法。

【請求項６】　クライアントコンピュータとサーバコンピュータとをネットワークを介して接続してなるクライアント・サーバシステムにおけるワークフローの変換方法において、
前記クライアントコンピュータにおいて、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換する段階と、
変換されたテキスト文字列をネットワークを通じて前記サーバコンピュータへ伝送する段階とを有し、
前記サーバコンピュータにおいて、前記ネットワークからテキスト文字列が受信された場合、変換対称ワークフローシステム用のワークフロー変換手段を起動する段階と、
起動されたワークフロー変換手段が、受信された文字列を、それぞれのワークフローシステム用のワークフローモデルに変換する段階と、
変換されたワークフローモデルを、それぞれのワークフローシステムに登録する段階とを有することを特徴とするワークフロー変換方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、例えば複数の異なるワークフローシステムを連携させるためのワークフロー定義方法、ワークフロー表示方法、ワークフロー再利用方法、ワークフロー変換方法に関する。

【０００２】

【従来の技術】従来から、ワークフロー管理システムとして、例えばInconcert（アメリカのTIBCO社商品名）等が良く知られている。

【０００３】このようなワークフロー管理システムは、人から人への仕事の受け渡しのルールをコンピュータ上に定義し、仕事の受け渡しを自動化するためのソフトウェアである。

【０００４】近年、上記以外にも様々なワークフロー製品が市場に存在しているが、ワークフロー定義（仕事の受け渡しのルール定義）の方法は、個々のワークフロー製品によって異なる。

【０００５】また、インターネットなどのネットワークを利用した業務処理が急速に普及している現在では、ワークフローの利用範囲も拡大される傾向にある。

【０００６】このようにワークフローの利用範囲が大きくなるに連れて、異なるワークフロー製品を統合する汎用的なワークフロー定義言語の必要が生じてきた。

【０００７】そこで、ワークフロー標準化団体WfMC(Workflow Management Coalition)が発足し、汎用的なワークフロー定義言語として、Workflow Process Definition Language：WPDLが定義された。

【０００８】

【発明が解決しようとする課題】しかしながら、WPDLが定義されたものの、このWPDLの言語処理プログラムを開発する負担などから、WPDLの実装を済ませているワークフロー製品は多くないのが現状である。

【０００９】したがって、WPDLが実装されたワークフロー製品と既存のワークフロー製品とのシステム間連携は今後とも必要であり、このような連携技術を構築することが急務になっている。

【００１０】本発明はこのような課題を解決するためになされたもので、異なるワークフローシステム間で容易にワークフローを連携させることのできるワークフロー定義方法、ワークフロー表示方法、ワークフロー再利用方法、ワークフロー変換方法を提供することを目的としている。

【００１１】

【課題を解決するための手段】上記目的を達成するために、請求項１記載の発明のワークフロー定義方法は、ワークフロー定義に関る操作用のアイコン部品を画面に表示する段階と、前記アイコン部品が前記画面上に配置されることでワークフローを定義する段階と、ワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Language で表記したテキスト文字列として出力する段階とを有することを特徴としている。

【００１２】請求項２記載の発明のワークフロー表示方法は、ワークフロー処理が実行されているシステムより、ワークフローの進捗状況の情報をExtensible Markup Language で表記したテキスト文字列で取得する段階と、取得されたテキスト文字列を、ワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの形態で表示する段階とを有することを特徴としている。

【００１３】請求項３記載の発明のワークフロー変換方法は、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列を解析する段階と、解析されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成する段階と、生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成する段階と、生成されたワークフローモデルを該当ワークフローシステムに登録する段階とを有することを特徴としている。

【００１４】請求項４記載の発明のワークフロー再利用方法は、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換する段階と、変換されたテキスト文字列を他のワークフローシステムへ伝送する段階と、伝送されてきたテキスト文字列を、自身のワークフローに変換する段階とを有することを特徴としている。

【００１５】請求項５記載の発明のワークフロー変換方法は、ワークフロー定義に関る操作用のアイコン部品を画面に表示する段階と、前記アイコン部品が前記画面上に配置されることでワークフローを定義する段階と、ワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Language で表記したテキスト文字列に変換する段階と、変換されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成する段階と、生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成する段階と、生成されたワークフローモデルを該当ワークフローシステムに登録する段階とを有することを特徴としている。

【００１６】請求項６記載の発明のワークフロー変換方法は、クライアントコンピュータとサーバコンピュータとをネットワークを介して接続してなるクライアント・サーバシステムにおけるワークフロー変換方法において、前記クライアントコンピュータにおいて、あるワークフローシステムのワークフローをExtensible MarkupLanguage で表記したテキスト文字列に変換する段階と、変換されたテキスト文字列をネットワークを通じて前記サーバコンピュータへ伝送する段階とを有し、前記サーバコンピュータにおいて、前記ネットワークからテキスト文字列が受信された場合、変換対称ワークフローシステム用のワークフロー変換手段を起動する段階と、起動されたワークフロー変換手段が、受信された文字列を、それぞれのワークフローシステム用のワークフローモデルに変換する段階と、変換されたワークフローモデルを、それぞれのワークフローシステムに登録する段階とを有することを特徴としている。

【００１７】請求項１記載の発明では、画面上に表示されたアイコン部品が画面上に配置されることでワークフローを定義し、このワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Language で表記したテキスト文字列として出力するので、テキストエディタ等を使わずとも、ワークフローの定義をExtensible Markup Language で表記できる。

【００１８】すなわち、画面上でのアイコン部品の配置操作、つまりドラッグアンドドロップの操作等でワークフローを定義でき、しかもこれと同時に、ワークフロー

定義のXML変換が自動的に行われるので、ユーザは、あるワークフローモデルについての知識さえあれば、そのワークフローモデルの表記法もXMLの表記法も知らなくても、XMLで表記したワークフローモデルを作成することができる。

【００１９】請求項２記載の発明では、ワークフロー処理が実行されているシステムより、ワークフローの進捗状況の情報をExtensible Markup Language で表記したテキスト文字列で取得し、取得されたテキスト文字列を、予め規定された文書型定義に基づき、ワークフローの形態で表示するので、ユーザは、現在のワークフローの進捗状況をビジュアル的に確認することができる。

【００２０】請求項３記載の発明では、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列を解析し、解析されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成し、この生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成し、生成されたワークフローモデルを該当ワークフローシステムに登録するので、異種ワークフロー間のワークフロー連携が可能になる。

【００２１】請求項４記載の発明では、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換し、この変換されたテキスト文字列を他のワークフローシステムへ伝送し、他のワークフローシステムでは、伝送されてきたテキスト文字列を、自身のワークフローに変換するので、一度、作成したワークフローモデルを他のワークフローシステムで再利用できるようになる。

【００２２】請求項５記載の発明では、画面上に表示されたアイコン部品が画面上に配置されることでワークフローを定義し、このワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Language で表記したテキスト文字列に変換するので、テキストエディタ等を使わずとも、ワークフローの定義をExtensible Markup Language で表記できる。

【００２３】また、変換されたテキスト文字列を解析し、解析されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成し、この生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成し、生成されたワークフローモデルを該当ワークフローシステムに登録するので、異種ワークフロー間のワークフロー連携が可能になる。

【００２４】請求項６記載の発明では、クライアントコンピュータにおいて、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換し、この変換されたテキスト文字列をネットワークを通じてサーバコンピュータへ伝送し、サーバコンピュータにおいてネットワークからテキスト文字列が受信された場合、変換対称ワークフローシステム用のワークフロー変換手段を起動し、起動されたワークフロー変換手段が、受信された文字列を、それぞれのワークフローシステム用のワークフローモデルに変換し、変換したワークフローモデルをそれぞれのワークフローシステムに登録するので、異なるワークフローシステム間のワークフロー同期をとることができる。

【００２５】この結果、異なるワークフローシステム間で容易にワークフローを連携させることができる。

【００２６】

【発明の実施の形態】以下、本発明の実施の形態を図面を参照して詳細に説明する。図１は本発明に係る一つの実施の形態のワークフロー連携論理モデル、図２は図１の論理モデルを具現化したワークフロー連携システムの一例を示す構成図である。

【００２７】図１において、符号１は異なるワークフロー間でやりとりされるWPDL-XMLである。このWPDL-XML１は、WPDL-XMLの文書構造や意味を表現するために使用するタグ情報が記述されたWPDL-DTDファイル１ａと、このWPDL-DTDファイル１ａに従って記述されたWPDL-XMLファイル１ｂとからなる。DTDファイルは、個々のXMLファイルがどのようなタグや属性を持つべきかという規定として定義するファイルであり、上記WPDL-DTDファイル１ａは、それをWPDLの定義に適合させたものである。 XMLとは、Extensible Markup Languageの略である。DTDとは、Document TypeDefinitionの略である。WPDL-XMLファイル１ｂはデータによってその内容が異なるが、WPDL-DTDファイルの内容は固定である。

【００２８】符号２～６はアプリケーションソフトウェアである。２はテキストエディタ(Windows付属のメモ帳など)である。テキストエディタ２は、テキストファイルであるWPDL-XMLファイル１ｂをWPDL-XMLの表記法に従って編集するためのツールである。３はWPDL-XMLワークフロー定義プログラムであり、ユーザがWPDL-XMLの表記法を知らなくてもGUI操作によりワークフローモデルの定義（ワークフローの流れの設定）を行え、この際、操作内容に応じてWPDL-XMLファイル１ｂを生成するプログラムである。４はWPDL-XMLワークフロー表示プログラムであり、読み込んだWPDL-XMLファイルを基にワークフロー情報をグラフィカルに表示するプログラムである。５はXMLパーサ/レンダラであり、XMLファイルの読み込み及び書き出しを行うプログラム（ソフトウェア）である。このプログラム（ソフトウェア）については市販のXMLパーサ/レンダラ製品を使用できる。６はXMLパーサであり、XMLファイルの読み込み、所定の形態、例えばワークフロー情報などに変換するプログラム（ソフトウ

ェア）である。このプログラム（ソフトウェア）も市販のXMLパーサ製品を使用できる。

【００２９】符号１０は業務システムであり、独立してワークフローを実行するワークフローシステム１１とWPDL-XMLアダプタ１２とを備えたシステムであり、ワークフローシステム１１を中心とし、この業務システム１０特有の業務データベース（以下業務ＤＢと称す）と業務アプリケーションとの連携を行う。WPDL-XMLアダプタ１２は、WPDL-XMLファイル１ｂを読み込み、ワークフローシステム１１への定義として登録し、また、ワークフローシステム１１のワークフロー進捗情報を、WPDL-XMLファイル１ｂに書き出す、このシステム対応用のアダプタである。WPDL-XMLアダプタ１２には、上記XMLパーサ/レンダラ５と同様のXMLパーサ/レンダラ１３が備えられている。

【００３０】符号２０は業務システムであり、独立してワークフローを実行するワークフローシステム２１とWPDL-XMLアダプタ２２とを備えたシステムであり、ワークフローシステム２１を中心とし、この業務システム２０特有の業務ＤＢと業務アプリケーションとの連携を行う。WPDL-XMLアダプタ２２は、WPDL-XMLファイル１ｂを読み込み、ワークフローシステム２１への定義として登録し、また、ワークフローシステム２１のワークフロー進捗情報を、WPDL-XMLファイル１ｂに書き出す、このシステム対応用のアダプタである。WPDL-XMLアダプタ２２には、上記XMLパーサ/レンダラ５と同様のXMLパーサ/レンダラ２３が備えられている。

【００３１】この図１の論理モデルを、クライアント・サーバシステムとして具体化した例が図２である。

【００３２】図２に示すように、この例のクライアント・サーバシステムは、インターネットに接続されたクライアントコンピュータ３１（以下クライアントＰＣ３１と称す）及びサーバ用コンピュータ３２と、このサーバ用コンピュータ３２とＬＡＮ等のイントラネット介して接続された個々にワークフロー処理を実行するワークフローサーバ用のコンピュータ３４、３７とからなり、クライアントＰＣ３１上のインターネットブラウザ４２からサーバ用コンピュータ３２にアクセスし、WPDL-XMLファイル１ｂを送受することで、各ワークフロー処理用のコンピュータ３４、３７とワークフローの連携を行うよう構成されている。

【００３３】コンピュータ３４には、ワークフローサーバ３５とワークフロー定義データを格納するデータベース３６とが設けられている。コンピュータ３７には、ワークフローサーバ３８とワークフロー定義データを格納するデータベース３９とが設けられている。

【００３４】クライアントＰＣ３１は、ＣＰＵ、ＲＯＭ、ＲＡＭ等のメモリ、ハードディスクドライブ装置等の補助記憶装置、モニタ、キーボード等の入出力装置を有するコンピュータであり、例えばWindowsやMacintosh等のオペレーティングシステム（以下ＯＳと称す）の環境下で動作するものである。ハードディスクドライブ装置には、WPDL-XML作成プログラム４１、インターネットブラウザ４２等のソフトウェアがインストールされている。WPDL-XML作成プログラム４１は、ワークフロー定義情報ファイルであるWPDL-XMLファイル４３を作成する。このWPDL-XMLファイル４３は、WPDL-XML定義プログラム４１の他、一般のテキストエディタで作成できる。インターネットブラウザ４２は、ブラウジング操作によりサーバ用コンピュータ３２のWWWサーバ５１に接続し、WWWサーバ５１に保持されているHTMLで作成されたファイル（HTMLファイル５９）をダウンロードしインターネットブラウザ４２の画面にワークフロー定義受付ページを表示する。このワークフロー定義受付ページから、ユーザが所定操作を行うことで、作成済みのWPDL-XMLファイル４３を電子メールに添付し、WWWサーバ５１に転送する。

【００３５】サーバ用コンピュータ３２は、例えばＵＮＩＸ（登録商標），WindowsＮＴ（登録商標）等のＯＳをインストールしたコンピュータであり、WWWサーバ５１、WWWサーバ側プログラム５２、WPDL-XMLアダプタ５３、５６、ワークフローサーバ３５用のクライアントライブラリ５５、ワークフローサーバ３８用のクライアントライブラリ５８等を有している。

【００３６】WWWサーバ５１は、HTTPリクエストを処理するサーバである。WWWサーバ側プログラム５２は、WWWサーバ５１へのリクエストに連動するように設定されたサーバ側プログラムである。サーバ側プログラムとしては、WWWサーバ５１のCGI(Common Gateway Interface)機能を使用しても良く、また、WWWサーバ５１に対応した市販のアプリケーションサーバ製品を使用して組み込んでも良い。いずれかの方法を用いることが一般的である。市販のアプリケーションサーバ製品としては、Ｊａｖａ（登録商標） Ｓｅｒｖｌｅｔ ＡＰＩ対応、Enterprise Java BeansAPI対応、CORBA対応などの様々な製品があり、必要とする機能によって製品を選定する。例えばWebsphere Application Server（IBM社製）、JRun Server（Allaire社製）、WebLogic Application Server（BEA社製）等、様々な製品が使用可能である。

【００３７】WWWサーバ側プログラム５２は、クライアントＰＣ３１から転送されてきたWPDL-XMLファイル４３を、サーバ用コンピュータ３２上の所定のディレクトリに格納し、サーバ用コンピュータ３２上に存在する該当WPDL-XMLアダプタ５３，５６の登録処理を起動する。WPDL-XMLファイル４３は、クライアントＰＣ３１から転送されたワークフロー定義情報である。

【００３８】WPDL-DTDファイル６０は、サーバ用コンピュータ３２に格納される文書型定義情報である。このWPDL-DTDファイル６０は、このようにインターネット／イントラネット上の所定のサーバに格納しておき、URL(Un

iform Resouce Locator)形式でクライアントＰＣ３１から参照するようにしても良く、また、個々のクライアントＰＣ３１に格納しても良い。

【００３９】WPDL-XMLアダプタ５３は、ワークフローシステム１１用のWPDL-XMLアダプタであり、XMLパーサ５４を有している。XMLパーサ５４は、WPDL-XMLアダプタ５３の内部で、WPDL-XMLファイル４３の解析を行うプログラムであり、市販の製品を利用できる。

【００４０】ワークフローシステム１１用のクライアントライブラリ５５は、ワークフローサーバ３５が提供しているクライアント用ライブラリである。WPDL-XMLアダプタ５３は、このワークフローシステム１１用のクライアントライブラリ５５の機能を使用することにより、WPDL-XMLデータをワークフローサーバ３５に適合させる。

【００４１】ワークフローサーバ３５は、ワークフロー処理を実行するサーバプログラムであり、ワークフローシステム１１用のクライアントライブラリ５５からのリクエストを受け付けてこのワークフローサーバ３５独自の方式でデータを登録する。

【００４２】WPDL-XMLアダプタ５６は、ワークフローシステム２１用のWPDL-XMLアダプタであり、XMLパーサ５７を有している。XMLパーサ５７は、WPDL-XMLアダプタ５６の内部で、WPDL-XMLファイル４３の解析を行うプログラムであり、市販の製品を利用できる。

【００４３】ワークフローシステム２１用のクライアントライブラリ５８は、ワークフローサーバ３８が提供しているクライアント用ライブラリである。WPDL-XMLアダプタ５６は、このワークフローシステム２１用のクライアントライブラリ５８の機能を使用することにより、WPDL-XMLデータをワークフローサーバ３８に適合させる。

【００４４】ワークフローサーバ３８は、ワークフロー処理を実行するサーバプログラムであり、ワークフローシステム２１用のクライアントライブラリ５８からのリクエストを受け付けて、このワークフローサーバ３８独自の方式でデータを登録する。

【００４５】図３は、図１の論理モデルに従い、サーバあるいはクライアントのコンピュータに実装したプログラムの一例を示す図である。

【００４６】ワークフローシステムのプログラムとしては、WPDL-XML InConcertアダプタ７１、InConcert Java連携ライブラリ７２、InConcertサーバ７３などである。

【００４７】WPDL-XML InConcertアダプタ７１は、WPDL-XMLのワークフロー定義データを、InConcertのワークフローデータとして登録し、また、InConcertのワークフローデータを、WPDL-XML形式で出力する、双方向変換アダプタである。このWPDL-XMLInConcertアダプタ７１には、IBM XML Parser for Java７４が使用されている。IBM XML Parser for Java７４は、XMLパーサ/レンダラである。

【００４８】InConcert Java連携ライブラリ７２は、Java言語から利用できるInConcertクライアントライブラリである。InConcertサーバ７３は、ワークフロー管理システムInConcertのサーバプログラムである。

【００４９】クライアント側のアプリケーションとしては、WPDL-XMLワークフロー定義プログラム７５、WPDL-XMLワークフロー表示プログラム７６等である。各プログラムには、IBM XML Parser for Java７４が使用されている。

【００５０】プログラム言語にはJavaが使用されている。WPDL-DTDファイル７５は、WPDLの文法を基にXMLでの表記法を決め、タグを定義したものである。WPDL-XMLワークフロー定義プログラム７５は、ユーザがWPDL-XMLの表記法を知らなくても、GUI操作（アイコン部品のドラッグアンドドロップ操作）によりワークフロー定義が行え、GUI操作に応じてWPDL-XMLファイルを自動生成する定義プログラムである。

【００５１】WPDL-XMLワークフロー表示プログラム７６は、WPDL-XMLのワークフロー情報をグラフィカルに表示するプログラムである。これらのプログラムは、Javaのアプレットで開発されているので、インターネットブラウザを利用してＷＷＷサーバ上にあるWPDL-XMLファイルを表示することができる。

【００５２】ここで、図４を参照し上記図３の実装プログラムをインターネットのシステムモデルとして実現した例について説明する。図４に示すように、サーバ用コンピュータ８０には、Java Servlet対応アプリケーションサーバ８１、ＷＷＷサーバ５１、WPDL-XML InConcertアダプタ７１、InConcert Java連携ライブラリ７２等を実装する。Java Servlet対応アプリケーションサーバ８１は、Java Servlet８２を有している。

【００５３】クライアントＰＣ９０には、HTMLファイルやWPDL-XMLファイルをダウンロードするインターネットブラウザ４２、WPDL-XMLワークフロー定義プログラム７５、WPDL-XMLワークフロー表示プログラム７６等を実装する。

【００５４】また、各コンピュータ８０，９０には、WPDL-DTDファイル７５を格納しておく。

【００５５】ワークフロー処理を実行するワークフロー処理用コンピュータ１００には、InConcertサーバ７３とワークフロー定義データを格納したデータベース１０１を実装する。

【００５６】以下、個別の実装例について説明する。まず、WPDLの表記法をXMLで記述するための表記法について説明する。WPDLでは、ワークフローの用語及び構造をWPDLワークフローモデルとして定義している。このワークフローモデルを記述するための文法を、WPDL表記法と呼ぶ。本発明では、ワークフローモデルをそのままに、XMLでワークフローモデルを記述できるように表記法を変更した。この表記方をWPDL-XML表記法と呼ぶことにす

る。

【００５７】WPDL表記法では、ワークフロー情報を以下のように表記する。

キーワード　　'識別子'

サブキーワード　"値の文字列"または値の予約語または'識別子の参照'

＊

＊．．．

END_キーワード

具体的には、下記（例１）のように表記される。

【００５８】

【数１】

```
(例1)
        PARTICIPANT             'Sales_Department'
              NAME              "Sales Department"
              DESCRIPTION       "is involved in handling sales orders"
              TYPE              ORGANISATIONAL_UNIT
        END_PARTICIPANT
```

【００５９】この（例１）の表記を、以下のルールで、XMLで表記する。

・キーワードは、同名のXMLエレメント（タグ）とする。

・キーワードの識別子は、ID属性とする。

・サブキーワードは、同名のXMLエレメントとする。

・予約語は、同名のXMLエレメントとする。

・サブキーワードの値が、文字列の場合、サブキーワードと同名のXMLエレメントの内容とする（ダブルクォーテーションでは囲まない）。

・サブキーワードの値が、予約語の場合、予約語と同名※のXMLエレメントを指定する。

・サブキーワードの値が、識別子の参照であり、かつ一つだけの場合、サブキーワードのIDREF属性の値とする。

・サブキーワードの値が、識別子の参照であり、かつ複数ある場合、サブキーワードのIDREFS属性の値とする。

この場合、上記（例１）のWPDLの表記は、下記（例２）のようにXMLで表記できる。

【００６０】

【数２】

```
(例2)
<PARTICIPANT ID="Sales_Department">
    <NAME>Sales Department</NAME>
    <DESCRIPTION>is involved in handling sales orders</DESCRIPTION>
    <TYPE><ORGANIZATIONAL_UNIT/></TYPE>
</PARTICIPANT>
```

【００６１】例外ルールとして、下記のものを設定する。

・キーワードACTIVITYのサブキーワードIMPLEMENTATIONの値となる予約語、NONE、APPLICATIONS、WORKFLOW、LOOPは、XMLエレメントとせず、IMPLEMENTATIONエレメントのTYPE属性として表記する。

・キーワードACTIVITYのサブキーワードSPLITの値となる予約語、AND、OR、XORは、XMLエレメントとせず、SPLITエレメントのTYPE属性として表記する。以下に、これらの例外を含んだ元のWPDL表記例（例３）と、WPDL-XMLの表記例（例４）を示す。

【００６２】

【数３】

(例３)

```
ACTIVITY              'MailRoom'
        NAME                  "Mail Room"
        IMPLEMENTATION  APPLICATIONS
                            TOOL_LIST
                                'scan_document'
                                'identify_document'
                                'send_document'
                                DESCRIPTION "sequential execution"
                            END_TOOL_LIST
        PERFORMER       'Mail_Room_Clerk'
        SPLIT           AND
END_ACTIVITY
```

(例４)

```
<ACTIVITY ID="MailRoom">
        <NAME>Mail Room</NAME>
        <IMPLEMENTATION TYPE="APPLICATIONS">
                <TOOL_LIST IDREFS="scan_document identify_document send_document">
                        <DESCRIPTION>sequential execution</DESCRIPTION>
                </TOOL_LIST>
        </IMPLEMENTATION>
        <PERFORMER IDREF="Mail_Room_Clerk"/>
        <SPLIT TYPE="AND" />
</ACTIVITY>
```

【００６３】続いて、WPDL-XMLワークフロー定義プログラム７５（ワークフロー定義プログラム）とWPDL-XMLワークフロー表示プログラム７６（ワークフロー表示プログラム）について説明する。WPDL-XMLは、前述のWPDL-XML表記法を理解することにより、Windowsのメモ帳のようなテキストエディタ（テキスト編集ソフト）で作成することができるが、これでは、WPDL-XML表記法を理解したものでなければ、WPDL-XMLを記述することができない。

【００６４】そこで、本発明では、ワークフロー定義プログラムとしてWPDL-XMLワークフロー定義プログラム７５を設けた。

【００６５】WPDL-XMLワークフロー定義プログラム７５は、図５に示すように、XMLパーサ/レンダラ１１０、データオブジェクト生成部１１２、ワークフロー定義画面表示部１１４、XML表示画面表示部１１６、変換処理部１１８等からなる。

【００６６】XMLパーサ/レンダラ１１０は、WPDL-XMLファイルを読み込み解析する。また、XMLパーサ/レンダラ１１０は、データオブジェクト生成部１１２で生成されたの内容（データオブジェクトクラスのインスタンス）に応じてXMLエレメントオブジェクトを作成する。さらに、XMLパーサ/レンダラ１１０は、ワークフロー定義画面で定義された内容を、WPDL-DTDファイルに基づきWPDL-XMLファイルとして書き出す。データオブジェクト生成部１１２は、解析されたデータから、Javaで利用するためのデータオブジェクトクラスのインスタンスを生成する。ワークフロー定義画面表示部１１４は、ワークフロー定義画面を表示する。

【００６７】WPDL-XMLワークフロー定義プログラム７５及びWPDL-XMLワークフロー表示プログラム７６では、データオブジェクトクラスのインスタンスのデータ構造がアイコンや矢印等で表示される。XML表示画面表示部１１６は、XMLパーサ/レンダラ１１０により変換（作成）されたXML文字列を表示する。変換処理部１１８は、XMLパーサ/レンダラ１１０により変換（作成）されたXML文字列をWPDLのファイルに変換する。

【００６８】クライアントＰＣ３１にインストールしたWPDL-XMLワークフロー定義プログラム７５を起動することで、図６に示すようなワークフロー定義画面が表示される。このワークフロー定義画面は、オブジェクトを個々に定義する定義画面１２１と、Flowタブの画面１２２とXMLタブの画面１２３等の２つを切替え表示する編集用画面とから構成されている。

【００６９】起動時は、ワークフロー定義画面には、Flowタブの画面１２２が表示されており、ツールバー部分に表示されたアイコンの中から例えばコンピュータのアイコン１２５等をユーザが選択し編集用画面１２２上に移動及び配置する操作（ドラッグアンドドロップ等の操作）を行い、この操作を繰り返し、さらに、ツールバー部分の矢印アイコン１２６で編集画面上のコンピュータのアイコン１２５を接続することによって、WPDLワークフローモデルに従ったワークフローが定義される。また、個々のコンピュータのアイコン１２５に対して定義画面１２１にて値（業務種別、名称、役職、役柄等）を設定することで、WPDL-XMLが自動作成される。

【００７０】ここで、ユーザがワークフロー定義画面のＸＭＬタブを選択操作すると、図７に示すように、WPDLワークフローモデルに従ったワークフロー定義をXMLで記述したテキストデータ（XML文字列）がXMLタブの画面１２３に表示される。このようなXML文字列は、ワークフロー定義の途中でも確認することができる。

【００７１】続いて、図８を参照してWPDL-XMLワークフロー定義プログラム７５によるWPDL-XMLファイル編集処理動作について説明する。ワークフロー定義画面にて、解析対象のWPDL-XMLファイルが指定されると、WPDL-XML

ワークフロー定義プログラム７５では、指定されたWPDL-XMLファイルがXMLパーサ／レンダラ１１０によって読み込まれ、読み込まれたWPDL-XMLデータが解析される。

【００７２】解析後、WPDL-XMLワークフロー定義プログラム７５では、その解析したWPDL-XMLデータは、修正、追加等の変更が可能なように、データオブジェクトクラスのインスタンスとして作成されて（図８のＳ１００１）、データオブジェクト生成部１１２に保持される（Ｓ１００２）。データオブジェクトクラスは、XMLエレメント（タグ）の名前に対応して予め作成されている。

【００７３】データオブジェクト生成部１１２に保持されたデータオブジェクトクラスのインスタンスは、ワークフロー定義画面表示部１１４に出力されてワークフロー定義画面上に、処理の流れを示すデータオブジェクトとして表示される（Ｓ１００３）。

【００７４】ここで、ユーザにより画面操作が行われると（Ｓ１００４）、その操作内容に沿ってワークフロー定義が編集されて、データオブジェクトクラスのインスタンスとしてデータオブジェクト生成部１１２に保持される（Ｓ１００２）。

【００７５】データオブジェクト部１１２のデータオブジェクトクラスのインスタンスは、W3Cの定義するXMLのエレメントオブジェクトクラスのインスタンスを生成し、XMLパーサ／レンダラ１１０に渡す。

【００７６】XMLパーサ／レンダラ１１０では、XMLレンダラ機能を使って、XMLのエレメントオブジェクトクラスのインスタンスを、WPDL-XMLファイルとして書き出す（Ｓ１００６）。

【００７７】この書き出し時に、WPDL-XMLファイルを変換処理部１１８に出力し、変換処理部１１８でフォーマットをWPDLの形式に変換することにより、WPDLの表示形式で書き出すようにもできる。

【００７８】なお、本実施形態では、XMLの解析及び生成のために、IBM XML Parser for Javaを使用したが、この製品は、W3C(World Wide Web Consortium)が標準化した、XML Java APIに準拠した製品である。したがって、XML Java APIに準拠した製品であれば、これ以外の製品を使用することもできる。

【００７９】すなわち、WPDL-XMLワークフロー定義プログラム７５をユーザが使用することで、ユーザは、WPDLワークフローモデルについての知識さえあれば、WPDLの表記法もWPDL-XMLの表記法も知らなくても、WPDL-XMLを作成することができる。

【００８０】次に、WPDL-XMLワークフロー表示プログラム７６について説明する。このWPDL-XMLワークフロー表示プログラム７６は、図９に示すように、XMLパーサ１３０、データオブジェクト生成部１３２、ワークフロー表示画面表示部１３４等からなる。XMLパーサ１３０は、最新のWPDL-XMLファイルを読み込み解析する。データオブジェクト部１３２は、画面表示のためのデータオブジェクトを保持する。ワークフロー表示画面表示部１３４は、実行中のワークフローの進捗状況を表示する。

【００８１】続いて、このWPDL-XMLワークフロー表示プログラム７６のワークフロー進捗状況表示動作について説明する。なお、処理動作そのものは図８のＳ１００１～Ｓ１００３までと同じである。

【００８２】クライアントＰＣ９０のインターネットブラウザの画面にて、ユーザによって、ワークフロー処理が実行されているシステムのワークフロー進捗状況情報を示すWPDL-XMLファイルが指定されると、WPDL-XMLワークフロー表示プログラム７６が起動し、指定されたWPDL-XMLファイルがXMLパーサ１３０によって、ワークフローサーバ側からインターネットを通じて読み込まれてそのWPDL-XMLデータが解析される。この解析にはWPDL-DTDファイルが利用される。

【００８３】解析後、そのWPDL-XMLデータは、データオブジェクトクラスのインスタンスとされて（図８のＳ１００１）、データオブジェクト生成部１３２に保持される（Ｓ１００２）。データオブジェクトクラス自体は、XMLエレメント（タグ）の名前に対応して予め作成されている。

【００８４】データオブジェクト生成部１３２に保持されたデータオブジェクトクラスのインスタンスは、ワークフロー定義画面表示部１３４に出力されて、図１０に示すようなワークフロー表示画面（Flow Viewer）上に、現在のワークフローの進捗状況を示すデータオブジェクトとして表示される（Ｓ１００３）。

【００８５】この例では、パソコンの購入申請のワークフローにおいて、ある書面は上長承認を却下されて再申請指示のボックスに入れられ、再申請指示待ちの状態になっていることが解る。また、他の書面は、上長承認、購入決裁、購買処理を通過し、登録のボックスと通知のボックスにそれぞれ入れられていることが解る。

【００８６】続いて、WPDL-XML InConcertアダプタ７１について説明する。WPDL-XML InConcertアダプタ７１は、XMLパーサ／レンダラ１４１、データオブジェクト生成部１４３、データ登録部１４５、データ取得部１４７等からなる。データオブジェクト生成部１４３は、Javaプログラムで生成されている。XMLパーサ／レンダラ１４１としては、WPDL-XMLの解析及び生成のために、ワークフロー定義プログラム７５と同様に、IBM XML Parser for Javaが使用されている。データオブジェクト生成部１４３は、XMLパーサ／レンダラ１４１により解析された情報を、データ登録部１４５で使うためのデータオブジェクトクラスのインスタンスに変換する。データ登録部１４５は、WPDLのワークフローモデルに従ったデータを、InConcertのワークフローモデルに変換してInConcertに登録する。データ取得部１４７は、InConcertから取得したデータを、WPDLのワークフローモデルに従

ったデータ形式に変換する。

【００８７】なお、InConcert Java連携ライブラリ７２は、InConcertのクライアント機能をJavaプログラムから利用するためのクラスライブラリである。

【００８８】続いて、このWPDL-XML InConcertアダプタ７１の動作を説明する。この場合、変換対象のWPDL-XMLファイルが指定されると、XMLパーサ／レンダラ１４１は、指定された変換対象のWPDL-XMLファイルを読み込み、解析する。そして、解析した情報をデータオブジェクト生成部１４３に渡す。

【００８９】データオブジェクト生成部１４３は、XMLパーサ／レンダラ１４１によって解析された情報が渡されると、その情報をデータ登録部１４５で使うためのデータオブジェクトクラスのインスタンスに変換しデータ登録部１４５に渡す。

【００９０】データ登録部１４５は、データオブジェクト生成部１４３よりデータオブジェクトクラスのインスタンスを受け取ると、InConcert Java連携ライブラリ７２を参照し、WPDLのワークフローモデルに従ったデータを、InConcertのワークフローモデルに変換してInConcertに登録する。

【００９１】一方、図４に示したシステム構成において、InConcertサーバ7.3におけるワークフローの進捗状況をクライアントＰＣ９０のユーザが確認するために、インターネットブラウザ４２からＷＷＷサーバ５１のワークフロー定義XML受付ページにアクセスし、表示されたワークフロー定義XML受付ページにて所定の操作を行うと、Java Servlet対応アプリケーションサーバ８１によってWPDL-XML InConcertアダプタ７１が起動される。

【００９２】すると、WPDL-XML InConcertアダプタ７１では、図１１に示したデータ取得部１４７がInConcertサーバ７３からデータを取得し、取得したデータをデータオブジェクト生成部１４３に渡す。

【００９３】データオブジェクト生成部１４３では、渡されたInConcertのデータをWPDLのワークフローモデルに従ったデータ形式に変換し、XMLパーサ／レンダラ１４１に渡す。

【００９４】XMLパーサ／レンダラ１４１は、データオブジェクト生成部１４３より渡されたWPDLのデータをXMLに変換してWPDL-XMLファイルとして出力する。

【００９５】すなわち、WPDL-XML InConcertアダプタ７１は、ワークフローシステムInConcertを対象に、WPDL-XMLのワークフロー定義情報を、InConcertのワークフロー定義情報として登録し、InConcertのワークフロー定義情報を、WPDL-XML形式で出力するものである。

【００９６】なお、WPDLのワークフローモデルは、既存のワークフローシステムが個別に使用しているワークフローモデルと同一とは限らない。用語や構造になんらかの差異があり得る。したがって、上記以外は、WPDL-XML InConcertアダプタ７１ではなく変換アダプタとなり、WPDLのワークフローモデルを、個別のワークフローシステムのモデルに変換し、また、その逆を行うプログラムとする。

【００９７】InConcertとWPDLとは、構造的に共通点が多く、主な変換は用語の違いと言える。したがって、変換プログラムは、比較的容易に開発できる。しかし、実行者の指定法、条件の指定法、サブフローの指定法など、構造が異なる部分もある。

【００９８】WPDLのワークフローモデルとInConcertのワークフローモデルとの間の差異と、その変換方法について説明する。

【００９９】双方向変換するにあたり、図１２に示すような事項を考慮する必要がある。

【０１００】この中でも、特に、考慮を要した４つの点について説明する。１．実行者の指定WPDLとInConcertの実行者指定の方法には、図１３のような違いがある。

【０１０１】例えばWPDLでは、作業単位であるACTIVITYのPERFORMER属性に対し、HUMAN（人）、ORGANIZATIONAL UNIT（部門）、ROLE（役割）、SYSTEM（アプリケーション）のいずれかのPARTICIPANTオブジェクトを割り当てるが、InConcertの場合には、作業単位Taskに対し、Role（役割）オブジェクトを割り当て、Roleの実行者として、Poolオブジェクトを割り当てる。Poolオブジェクトはuser（利用者）の集合である。

【０１０２】この２つの異なるモデルを、以下のように変換する。

【０１０３】（WPDL→InConcert）

・HUMAN： HUMANエレメントは人を表すので、Userに相当する。しかしInConcertではUserを含むPoolを作成しなければならないので、UserとUserを含むPoolの両方を作成する。

・ORGANIZATIONAL_UNIT： ORGANIZATIONAL_UNITエレメントは部門を表す。InConcertではPoolがこれに相当する。ORGANIZATIONAL_UNITに所属する利用者のリストが定義時に得られれば、その数だけUserを作成し、Poolに割り当てる。

・ROLE： ROLEエレメントは役割を表す。Roleオブジェクトを作成して、Taskに割り当てる。

・SYSTEM： SYSTEMエレメントはアプリケーションの実行を表す。InConcertでは、Taskに直接アプリケーションを指定することはできないが、Taskの開始時のイベントをキャッチして特定のアプリケーションに起動をかける設定をすることができる。これを利用し、SYSTEMエレメントで指定されたアプリケーションを起動するAction Specオブジェクトを作成。次に、タスク開始イベントから、これを実行するTriggerオブジェクトを作成する。

【０１０４】（InConcert→WPDL）

・Pool： InConcertのPoolオブジェクトは、HUMANエレメントに相当する場合と、ORGANIZASIONAL_UNITエレメ

ントに相当する場合が考えられる。この判別方法として、Poolに所属するUserが単一のときはHUMANエレメント、複数のときはORGANIZATIONAL_UNITエレメントを作成する。

・Role： ROLEエレメントを作成する。

・Trigger： InConcertのTaskにTriggerが設定されている場合は、起動されるActionSpecオブジェクトから、アプリケーション名を取得し、SYSTEMエレメントを作成する。

【０１０５】２．条件の指定

図１７に示すように、ワークフローでの条件は、ある作業単位から別の作業単位に遷移するときに設定される。WPDLでの条件は、分岐条件であり、ある作業単位の次のルートが複数ある場合、条件判断を行って、通過するルートを決める。条件に合致しなかったルートは、通らない。InConcertでの条件は、ルートを選択する条件ではなく、作業の実行条件である。存在するルートは、無条件に通過し、作業単位に到達してから、条件判断を行い、作業単位を実行するかどうかを決める。

【０１０６】変換時には、それぞれの実行条件を、拡張属性として保持する。

【０１０７】・WPDL→InConcert WPDL_CONDITION属性の値として保持する。

・InConcert→WPDL EXTENDED_ATTRIBUTEエレメントのPERFORM_CONDITIONの値として保持する。

【０１０８】３．サブフローの指定

WPDLもInConcertも、ワークフローの作業の一部として、別のワークフローを指定することができる。

【０１０９】例えば図１５に示すように、WPDLでは、並列した複数のワークフローを記述して、一方のワークフローの作業単位(ACTIVITY)で、もう一方のサブフローとして指定する。InConcertでは、作業単位（Task）を階層構造にすることにより、サブフローの指定が可能である。

【０１１０】４．拡張属性の扱い

WPDLでは、ワークフロープロセスに任意の拡張属性を追加できる。これをInConcertに登録するときに以下の処理を行う。

【０１１１】WPDL情報を登録するWPDLクラスに、属性名を格納する属性と、属性値を格納する属性を予め用意する。この組を、整数型、文字列型、日付型の３種類用意する。つまり、以下の６通りである。

【０１１２】・整数属性名を格納する属性（文字列型）
・整数属性値を格納する属性（整数型）
・文字列属性名を格納する属性（文字列型）
・文字列属性値を格納する属性（文字列型）
・日付型属性名を格納する属性（文字列型）
・日付型属性値を格納する属性（日付型）

複数の属性を登録できるように、この６通りをそれぞれ複数個用意しておく。WPDL-XMLにより与えられた属性がその数を上回ったときはその都度属性の組を追加する。

【０１１３】ここで、図１６を参照してWPDL-XMLの拡張属性をInConcertに登録する場合の一例について説明する。

【０１１４】WPDL-XMLでは、ワークフロープロセスに任意の拡張子を追加できるが、これをそのままInConcertに登録すると、InConcert側では、変換のたびにデータベース上に新しい属性を追加する処理を実行することになりこの際の負荷が重くなる。

【０１１５】そこで、本実施形態の場合、図１６に示すように、InConcert側に予め属性名と属性値を格納するための属性（テンプレート）１５１を用意しておく。

【０１１６】例えば登録対象のWPDL-XML１５０が、図のような名前（STR_A, STR_B, INT_A）、値（“AA”，“BB”，５）、型（STRING, STRING, INTEGER）を持っていた場合、この属性（テンプレート）１５１を使用することによって、行を１行づつ追加しながら、それぞれの名前、値、型を当てはめるだけで良くなり、InConcert側で変換のたびにデータベース上に新しい属性を追加する処理の負荷を軽減することができる。

【０１１７】続いて、独立したワークフローシステム間の連携モデルについて説明する。図２で示した、インターネットにおけるシステムモデルは、一つのWWWサーバの管理下に、複数の異なるワークフローシステムが存在するような場合、例えば、企業のメインのホームページのバックエンドに、部門ごとに異なるワークフローサーバが動作している場合や、グループ企業のメインのホームページのバックエンドに、企業ごとに異なるワークフローサーバが動作している場合などに有効である。

【０１１８】これに対し、システムを共有しない企業間の異なるワークフローシステムを連携させたい場合がある。この場合、システム間をつなぐ通信方式（メール、メッセージング、分散オブジェクトなど）が必要になる。

【０１１９】複数のワークフローサーバのいずれか、またはすべてが、WWWサーバの管理下にない場合は、図２の構成を一部拡張し、図１７に示すような構成をとる必要がある。この例では、ワークフローシステムが、WWWサーバ５１の管理下にはない独立したシステムであるような場合を想定している。

【０１２０】すなわち、この例の場合、サーバ用コンピュータ３２に、通信プログラム８３を新たに設ける一方、WWWサーバ５１の管理下にはない独立したワークフローシステム１６０に通信プログラム１６１、WPDL-XMLアダプタ５７及びワークフローサーバ３８用のクライアントライブラリ５８を設ける。

【０１２１】通信プログラム８３は、ワークフローシステム１６０に対し、WPDL-XMLデータの転送を行う通信プログラムである。通信プログラム８３の実装形態としては、分散オブジェクト、メッセージングシステム、メー

ル等、いくつかの方法が考えられる。

【０１２２】例えば分散オブジェクトの場合は、CORBAなどの分散オブジェクト製品を使用してWPDL-XMLデータを転送する。メッセージングシステムの場合は、非同期の通信が可能なメッセージング製品を使用してWPDL-XMLデータを転送する。メールの場合は、メールの添付ファイルを使用してWPDL-XMLデータを転送する。

【０１２３】通信プログラム（受信側）１６１は、転送されてきたWPDL-XMLデータを所定のディレクトリにファイルとして書き出す。WPDL-XMLファイル４３は、転送されてきたワークフロー定義情報ファイルである。WPDL-DTDファイル６０は、ワークフローシステム１６０とそのコンピュータ３７に配置されている。なお、DTDファイルは各コンピュータ３７に配置せず、インターネット上の特定に位置に配置し、URL形式で参照するようにしても良い。WPDL-XMLアダプタ５７は、ワークフローシステム１６０のためのWPDL-XMLアダプタである。XMLパーサ５４は、WPDL-XMLアダプタ５７の内部で、WPDL-XMLファイルの解析に使用されるプログラムである。これは、市販のXMLパーサ製品を利用できる。

【０１２４】ワークフローシステム１６０用のクライアントライブラリ５８は、ワークフローシステム１６０が提供しているクライアント用ライブラリである。WPDL-XMLアダプタ５７は、このライブラリの機能を使用することにより、WPDL-XMLデータをワークフローシステム１６０に適合させる。

【０１２５】ワークフローシステム１６０用のサーバ３８は、ワークフローシステム１６０のサーバプログラムである。リクエストを受け付けて、製品独自の方式でデータを登録する。

【０１２６】このワークフロー連携システムの場合、サーバ用コンピュータ３２の通信プログラム８３と、独立したワークフローシステム１６０の通信プログラム１６１間でWPDL-XMLファイルのやりとり（転送）を行うことで、独立したワークフローシステム１６０であってもワークフロー→ＸＭＬ、ＸＭＬ→ワークフロー等の双方向変換が可能になり、ワークフロー連携を行うことができる。

【０１２７】ここで、WPDLの表記法を、XMLで記述するための表記法の別の表記ルールについて説明する。

【０１２８】今回の実施形態は、WPDLの表記法から、極力少ないルールで、XMLの表記法に変換することを目的としている。したがって、ごくわずかの例外をのぞいて、すべてのWPDLキーワードやサブキーワードはXMLエレメント（タグ）として定義した。このことにより、変換作業を単純な作業にすることができる。

【０１２９】しかし、WPDLのワークフローモデルの意味を詳細に調査すると、サブキーワードのいくつかはXMLエレメントではなく、むしろエレメントの属性として定義した方が適当な場合があることも事実である。

【０１３０】例えば個々の仕事(Activity)の名前は、現在<NAME>タグによって記述される。

<ACTIVITY>

<NAME>申請書提出</NAME>

</ACTIVITY>しかし、一個のエレメントに必ず一つ現れるようなデータは、属性で定義した方が自然である。

【０１３１】<ACTIVITY NAME=" 申請書提出" />このようにWPDLワークフローモデルの意味を重視して、エレメントと属性を使い分ける表記法も考えられる。

【０１３２】WPDL-XMLワークフロー定義プログラム、WPDL-XMLワークフロー表示プログラム、WPDL-XML InConcert変換アダプタ等の各プログラムは、本実施形態において、Java言語によって作成されているが、この他、例えばWindows上のVisualC++、VisualBasic等でも作成できる。

【０１３３】このようにこの実施形態のワークフロー連携システムによれば、汎用的なワークフロー定義であるWPDLをXMLで記述したWPDL-XMLファイルを、インターネットあるいはイントラネットを介して接続された異なるワークフローシステム間でやりとりすることでワークフロー連携を容易に行うことができる。

【０１３４】また、WPDLの文法をXMLの表記法で忠実に再定義し、文書型定義を記述したDTDファイルを作成したので、WPDLをXMLで表記することが可能になった。これにより、市場に広く出回っている任意のXMLパーサ(XML文書構造を解析するプログラム)を使用して言語処理を行えるようになり、XMLを使わない場合と比較して開発の工数を削減することができる。

【０１３５】また、ユーザは、ワークフロー定義画面に表示されたアイコンをドラッグアンドドロップするというＧＵＩ操作でワークフローモデルを作成（表記）できる。しかも、そのとき作成したワークフローモデルがXMLの表記に自動的に変換されるので、ユーザは、WPDLのワークフローモデルについての知識さえあれば、WPDLの表記法もXMLの表記法も知らなくても、テキストエディタ等でキー入力あるいは編集操作することなくXMLで表記したワークフローモデルを容易に作成することができる。

【０１３６】さらに、XMLによるWPDL対応をそれぞれのワークフローシステムが行えば、異なるワークフローシステムが共通の定義データとアプリケーションプログラムを利用できるようになる。

【０１３７】なお、本発明は上記実施形態のみに限定されるものではない。上記実施形態に記載の各プログラム（ソフトウェア）は、フロッピー（登録商標）ディスクなどのコンピュータが読み出し可能な記憶媒体に記憶されていても良く、この場合、記憶媒体に記憶されたプログラム（ソフトウェア）をコンピュータが読み出すことにより、各実施形態における処理が可能になる。

【０１３８】なお、本発明における記憶媒体としては、

磁気ディスク、フロッピーディスク、ハードディスク、光ディスク（ＣＤ－ＲＯＭ、ＣＤ－Ｒ、ＤＶＤなど）、光磁気ディスク（ＭＯなど）、半導体メモリなど、プログラムを記憶でき、かつコンピュータが読み取り可能な記憶媒体であれば、その記憶形式はいずれの形態であっても良い。

【０１３９】また、記憶媒体からコンピュータにインストールされたプログラムの指示に基づき、コンピュータ上で稼動しているＯＳ（オペレーティングシステム）や、データベース管理ソフト、ネットワークソフトなどのＭＷ（ミドルウェア）などが本実施形態を実現するための各処理の一部を実行しても良い。

【０１４０】さらに、本発明における記憶媒体は、コンピュータと独立した媒体に限らず、ＬＡＮやインターネットなどにより伝送されたプログラムをダウンロードして記憶または一時記憶した記憶媒体も含まれる。

【０１４１】また、記憶媒体は一つに限らず、複数の媒体から本実施形態における処理が実行される場合も本発明における記録媒体に含まれ、媒体構成はいずれの構成であっても良い。

【０１４２】なお、本発明におけるコンピュータは、記憶媒体に記憶されたプログラムに基づき、本実施形態における各処理を実行するものであって、パソコンなどの一つからなる装置、複数の装置がネットワーク接続されたシステムなどのいずれの構成であっても良い。

【０１４３】また、本発明におけるコンピュータとは、パソコンに限らず、情報処理機器に含まれる演算処理装置、マイコンなども含み、プログラムによって本発明の機能を実現することが可能な機器、装置を総称している。

【０１４４】

【発明の効果】以上説明したように、請求項１記載の発明によれば、ユーザは、画面上でのアイコン部品の配置操作、つまりドラッグアンドドロップの操作等でワークフローを定義でき、しかもこれと同時に、ワークフロー定義のXML変換が自動的に行われるので、ユーザは、あるワークフローモデルについての知識さえあれば、そのワークフロー定義の表記法もXMLの表記法も知らなくても、XMLで表記したワークフローモデルを作成することができる。

【０１４５】請求項２記載の発明によれば、ワークフロー処理が実行されているシステムより、ワークフローの進捗状況の情報をExtensible Markup Language で表記したテキスト文字列で取得し、取得されたテキスト文字列を、予め規定された文書型定義に基づき、ワークフローの形態で表示するので、ユーザは、現在のワークフローの進捗状況をビジュアル的に確認することができる。

【０１４６】請求項３記載の発明によれば、ワークフローをExtensible Markup Languageで表記したテキスト文字列を解析し、ワークフロー連携ライブラリを参照することで、他のワークフローモデルを生成し、生成されたワークフローモデルを該当ワークフローシステムに登録するので、異種ワークフロー間のワークフロー連携が可能になる。

【０１４７】請求項４記載の発明によれば、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換し、この変換されたテキスト文字列を他のワークフローシステムへ伝送し、他のワークフローシステムでは、伝送されてきたテキスト文字列を、自身のワークフローに変換するので、一度、作成したワークフローモデルを他のワークフローシステムで再利用できるようになる。

【０１４８】請求項５記載の発明によれば、画面上に表示されたアイコン部品が画面上に配置されることでワークフローを定義し、このワークフローの定義をExtensible Markup Language で表記するために規定した文書型定義に基づき、ワークフローの定義内容をExtensible Markup Language で表記したテキスト文字列に変換するので、テキストエディタ等を使わずとも、ワークフローの定義をExtensible Markup Language で表記できる。

【０１４９】また、変換されたテキスト文字列を解析し、解析されたテキスト文字列を基に、所定言語のプログラムで編集可能なようにデータオブジェクトクラスのインスタンスを生成し、この生成されたデータオブジェクトクラスのインスタンスとワークフロー連携用ライブラリとに基づき、他のワークフローシステムのワークフローモデルを生成し、生成されたワークフローモデルを該当ワークフローシステムに登録するので、異種ワークフロー間のワークフロー連携が可能になる。

【０１５０】請求項６記載の発明によれば、クライアントコンピュータにおいて、あるワークフローシステムのワークフローをExtensible Markup Language で表記したテキスト文字列に変換し、この変換されたテキスト文字列をネットワークを通じてサーバコンピュータへ伝送し、サーバコンピュータにおいてネットワークからテキスト文字列が受信された場合、変換対称ワークフローシステム用のワークフロー変換手段を起動し、起動されたワークフロー変換手段が、受信された文字列を、それぞれのワークフローシステム用のワークフローモデルに変換し、変換したワークフローモデルをそれぞれのワークフローシステムに登録するので、異なるワークフローシステム間のワークフロー同期をとることができる。

【０１５１】この結果、異なるワークフローシステム間で容易にワークフローを連携させることができる。

【図面の簡単な説明】

【図１】本発明に係る一つの実施の形態のワークフロー連携モデルを示す図。

【図２】図１のワークフロー連携モデルを具現化したワークフロー連携システムの構成例を示す図。

【図３】図１の論理モデルに従い、サーバあるいはクラ

イアントのコンピュータに実装したプログラムの一例を示す図。

【図４】図３のプログラムをインターネットのシステムモデルとした一例を示す図。

【図５】WPDL-XMLワークフロー定義プログラムの構成例を示す図。

【図６】ワークフロー定義画面の一例を示す図。

【図７】ワークフロー定義画面にて、ワークフロー定義をXMLで表示させた例を示す図。

【図８】ワークフロー定義処理及び表示処理を示すフローチャート。

【図９】WPDL-XMLワークフロー表示プログラムの構成例を示す図。

【図１０】ワークフローの進捗状況を表示した一例を示す図。

【図１１】WPDL-XML InConcertアダプタの構成を示す図。

【図１２】WPDLのワークフローモデルとInConcertのワークフローモデルとの間の差異で変換に考慮する事項を示す図。

【図１３】WPDLとInConcertとの実行者指定方法の違いを示す図。

【図１４】WPDLとInConcertとの条件の指定方法の違い＊を示す図。

【図１５】WPDLとInConcertとのサブフローの指定方法の違いを示す図。

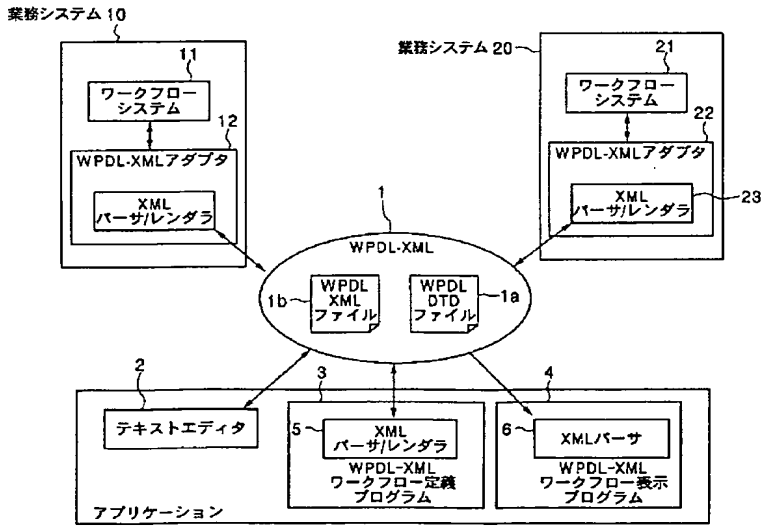
【図１６】WPDL-XMLの拡張属性をInConcertに登録する場合の一例を示す図。

【図１７】独立したワークフローシステム間の連携モデルを示す図。

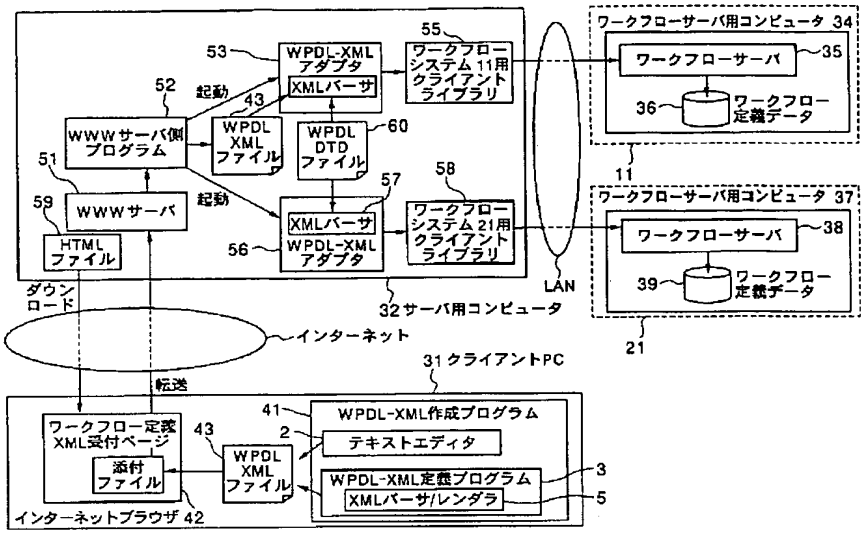
【符号の説明】

１…WPDL-XML、１ａ…WPDL-DTDファイル、１ｂ…WPDL-XMLファイル、２…テキストエディタ、３…WPDL-XMLワークフロー定義プログラム、４…WPDL-XMLワークフロー表示プログラム、５，１３，２３… XMLパーサ／レンダラ、６…XMLパーサ、１０，２０…業務システム、１１，２１…ワークフローシステム、１２，２２…WPDL-XMLアダプタ、５１…ＷＷＷサーバ、７１…WPDL-XML InConcertアダプタ、７２…InConcert Java連携ライブラリ、７５…WPDL-XMLワークフロー定義プログラム、７６…WPDL-XMLワークフロー表示プログラム、１４１…XMLパーサ／レンダラ、１４３…データオブジェクト生成部、１４５…データ登録部、１４７…データ取得部、１６０…ワークフローシステム、８３，１６１…通信プログラム。

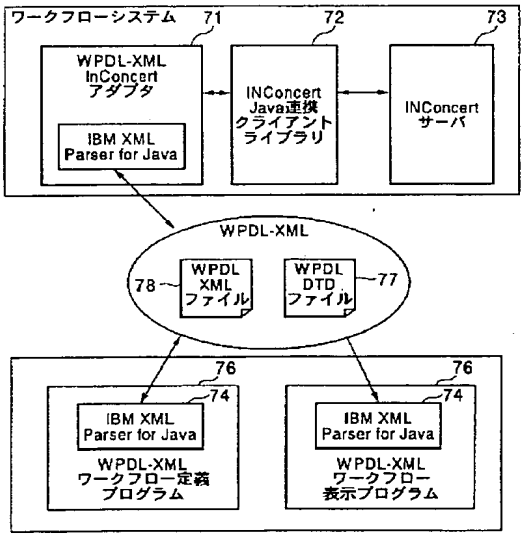
【図１】



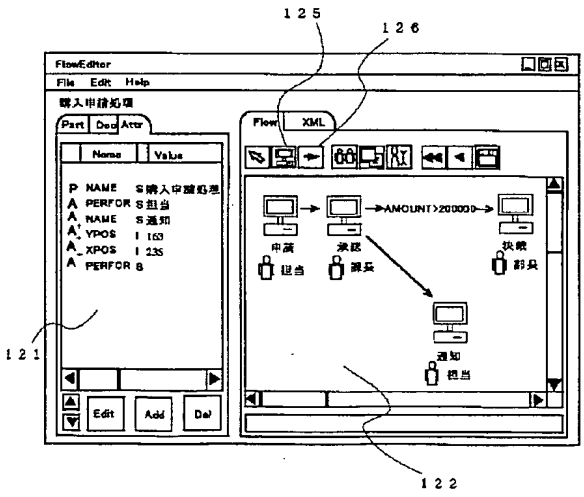
【図２】


ワークフローサーバ用コンピュータ 34
ワークフローサーバ
35
ワークフロー定義データ
36
11
ワークフローサーバ用コンピュータ 37
ワークフローサーバ
38
ワークフロー定義データ
39
21
LAN
53
WPDL-XML アダプタ
XMLパーサ
55
ワークフローシステム 11用クライアントライブラリ
52
起動
43
WWWサーバ側プログラム
WPDL XML ファイル
WPDL DTD ファイル
60
51
57
58
59
WWWサーバ
起動
XMLパーサ
ワークフローシステム 21用クライアントライブラリ
HTML ファイル
56
WPDL-XML アダプタ
ダウンロード
32サーバ用コンピュータ
インターネット
31 クライアントPC
転送
41
WPDL-XML作成プログラム
2
テキストエディタ
ワークフロー定義 XML受付ページ
43
WPDL XML ファイル
WPDL-XML定義プログラム
3
添付ファイル
XMLパーサ/レンダラ
5
インターネットブラウザ 42

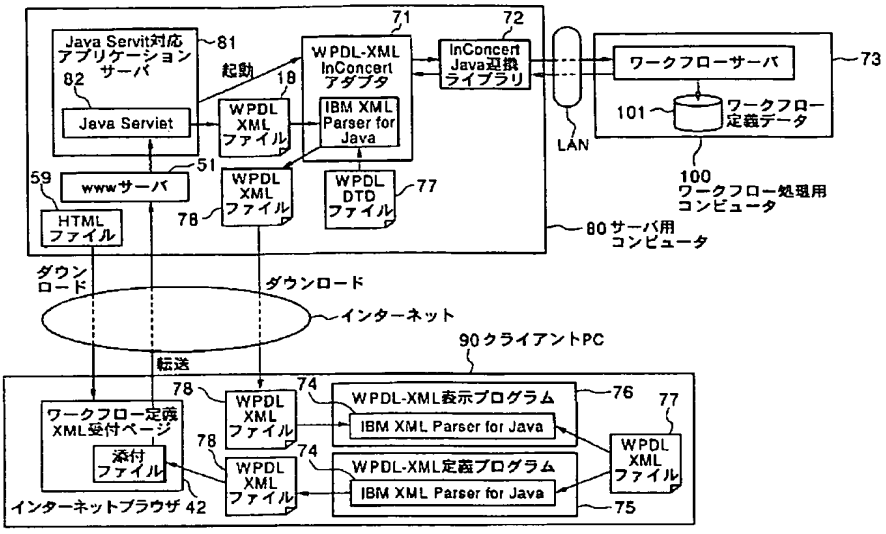
【図３】


ワークフローシステム
71
72
73
WPDL-XML InConcert アダプタ
IBM XML Parser for Java
INConcert Java連携クライアントライブラリ
INConcert サーバ
WPDL-XML
78
WPDL XML ファイル
WPDL DTD ファイル
77
76
74
IBM XML Parser for Java
WPDL-XML ワークフロー定義プログラム
IBM XML Parser for Java
WPDL-XML ワークフロー表示プログラム

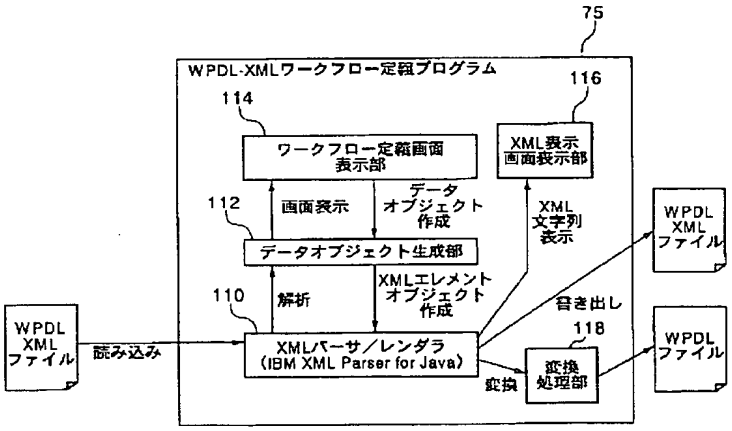
【図６】


125
126
FlowEditor
File Edit Help
購入申請処理
Part Data Actor
Name Value
Edit
Add
Del
Flow XML
121
122

【図４】



【図５】



【図１６】

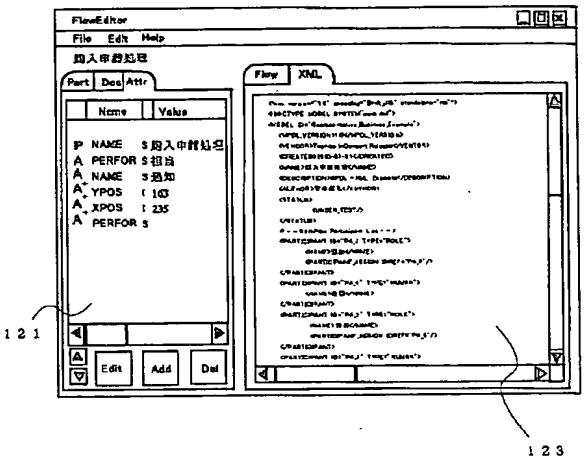
WPDL-XML 150

名前	値	型
STR_A	"AA"	STRING
STR_B	"BB"	STRING
INT_A	5	INTEGER

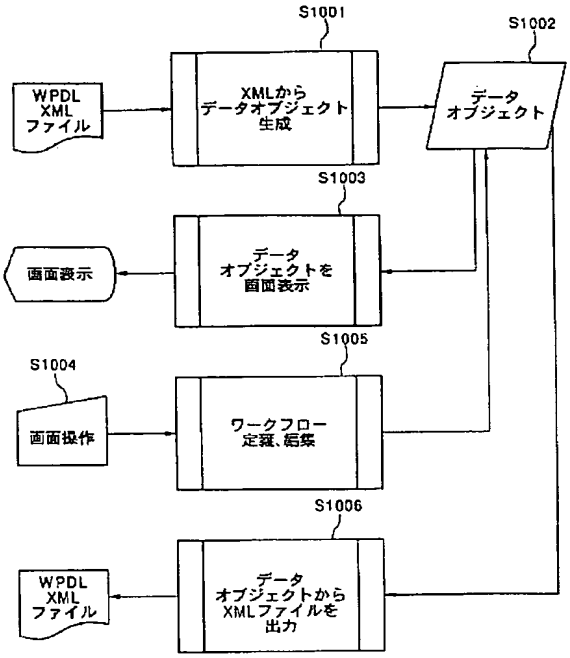
InConcert 151

名前	値	型
XEOSS_S_NAME1	STR_A	STRING
XEOSS_S_VALUE1	"AA"	STRING
XEOSS_S_NAME2	STR_B	STRING
XEOSS_S_VALUE2	"BB"	STRING
XEOSS_I_NAME1	INT_A	STRING
XEOSS_I_VALUE1	5	INTEGER

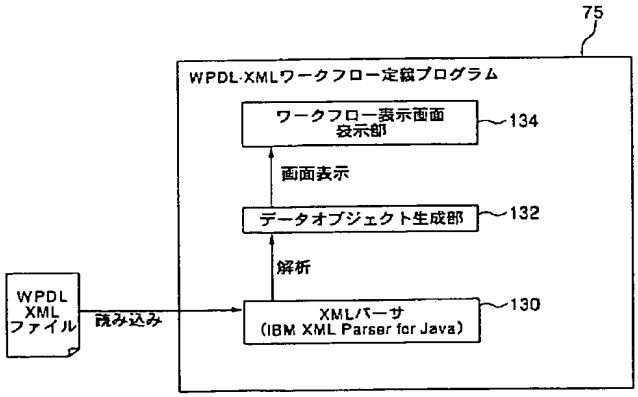
【図７】


FlowEditor
File Edit Help
Part Doc Attr
Name Value
Flow XML
Edit
Add
Del
121
123

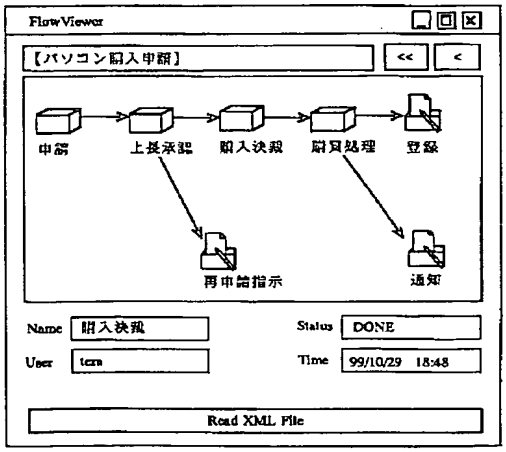
【図８】


S1001
S1002
WPDL XML ファイル
XMLから データオブジェクト 生成
データ オブジェクト
S1003
画面表示
データ オブジェクトを 画面表示
S1004
S1005
画面操作
ワークフロー 定義、編集
S1006
WPDL XML ファイル
データ オブジェクトから XMLファイルを 出力

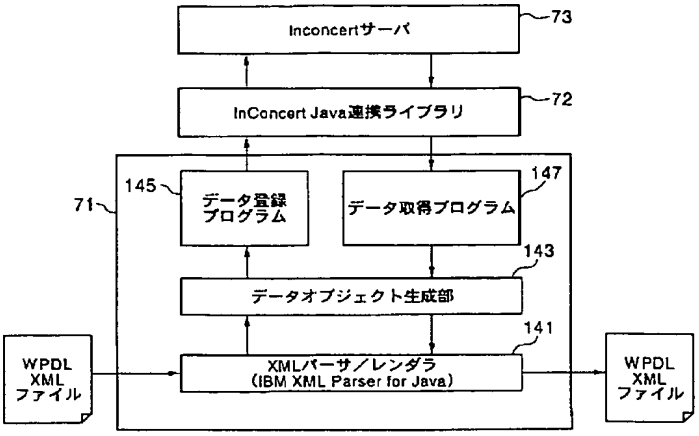
【図９】


75
WPDL-XMLワークフロー定義プログラム
ワークフロー表示画面 表示部
134
画面表示
データオブジェクト生成部
132
解析
WPDL XML ファイル
読み込み
XMLパーサ (IBM XML Parser for Java)
130

【図１０】


FlowViewer
【パソコン購入申請】
<<
<
申請
上長承認
購入決裁
購買処理
登録
再申請指示
通知
Name 購入決裁
Status DONE
User tom
Time 99/10/29 18:48
Read XML File

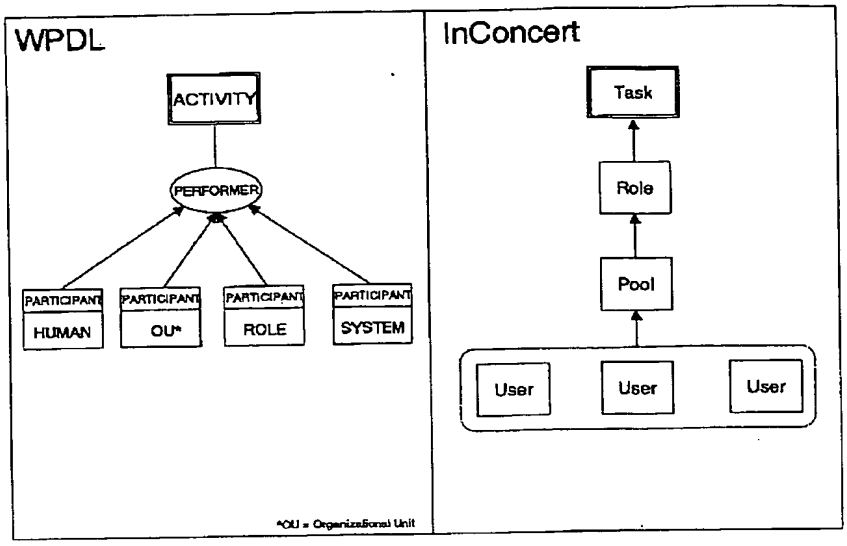
【図１１】



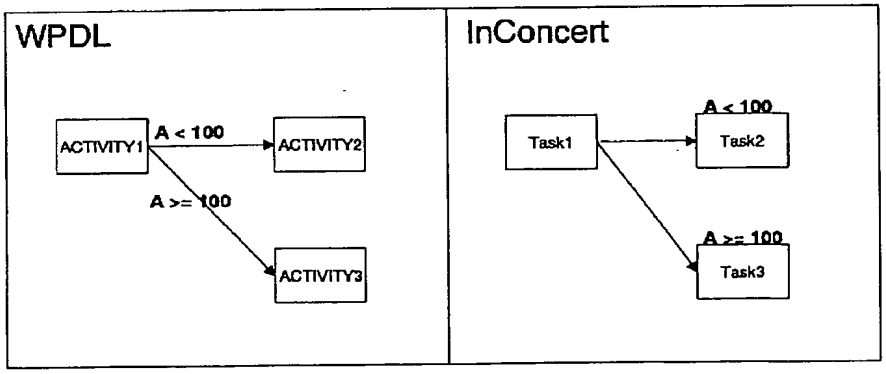
【図１２】

用語/用法	WPDL-XML	InConcert	WPDL→InConcert	InConcert→WPDL
ワークフロー	WORKFLOW	Job	WPDL-XML情報を格納できるWPDLクラスを作成し、WPDLクラスを継承するJobテンプレートとして定義。	WORKFLOWエレメントを作成
個々の作業	ACTIVITY	Task	Taskオブジェクトを作成	ACTIVITYエレメントを作成
作業の順序	TRANSITION	Link	Linkオブジェクトを作成。	TRANSITIONエレメントを作成
作業者	PARTICIPANT HUMAN	User Pool	同名のUserとPoolを作成し、UserをPoolに割り当てる	Userがひとつだけ割り当てられているPoolをHUMANエレメントとして作成。
部門	PARTICIPANT ORGANIZATIONAL UNIT	Pool	部門を表すPoolを作成し、部門に所属するUserを割り当てる	複数Userが所属するPoolをORGANIZATIONAL UNITエレメントとして作成する。
システム	PARTICIPANT SYSTEM	Trigger	Taskの開始に伴ってアプリケーションを起動するTriggerオブジェクトを作成	Triggerの対象アプリケーションをSYSTEMエレメントとして作成
役割	PARTICIPANT ROLE	Role	Roleオブジェクトを作成。	ROLEエレメント作成。
文書	DATA	Document	Documentオブジェクト作成	DATAエレメント作成。
条件	CONDITION	PerformCondition	WPDLクラスのWPDL_CONDITION属性の値として登録	PERFORM_CONDITIONという拡張属性の値として登録。
サブフロー	ACTIVITY が参照するWORKFLOW	Task(階層構造)	ACTIVITYエレメントに対応するTaskのサブTaskとして登録	別のWORKFLOWエレメントとして作成し、Taskに対応するACTIVITYエレメントから参照

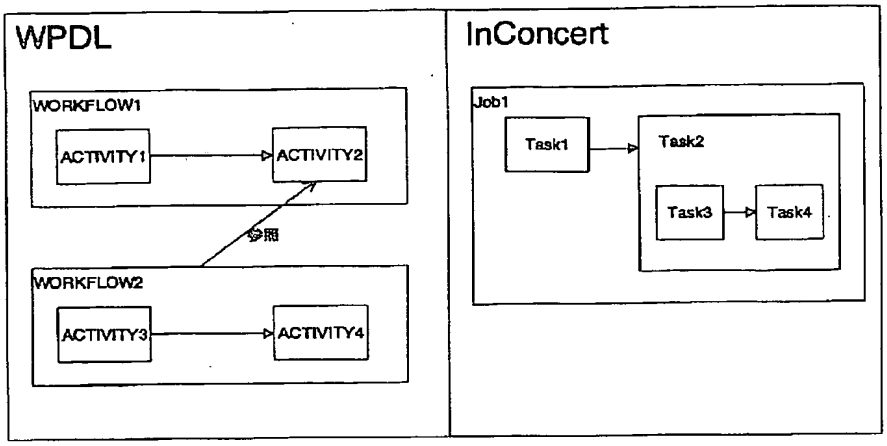
【図１３】


WPDL
ACTIVITY
PERFORMER
PARTICIPANT
HUMAN
PARTICIPANT
OU*
PARTICIPANT
ROLE
PARTICIPANT
SYSTEM
*OU = Organizational Unit
InConcert
Task
Role
Pool
User
User
User

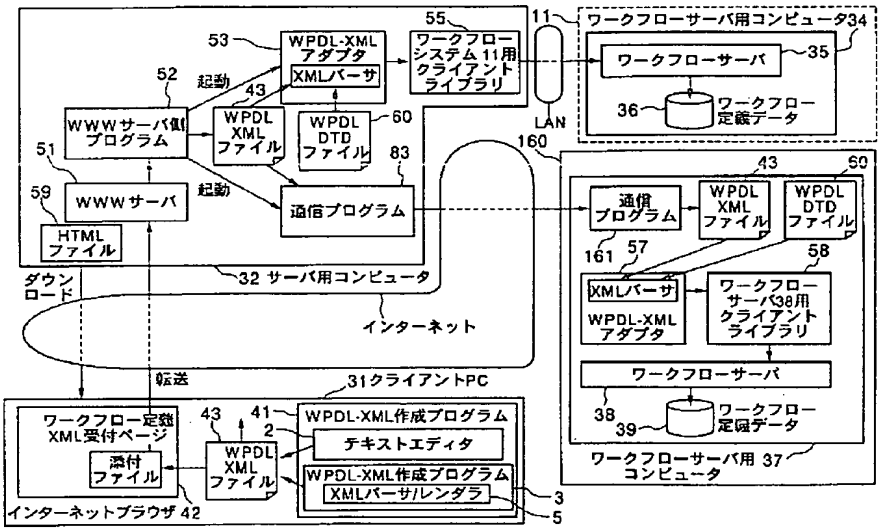
【図１４】


WPDL
ACTIVITY1
A < 100
ACTIVITY2
A >= 100
ACTIVITY3
InConcert
Task1
A < 100
Task2
A >= 100
Task3

【図１５】



【図１７】



フロントページの続き

(72)発明者　吉本　武弘
東京都港区芝浦１丁目１番１号　東芝ビルディング　東芝システム開発株式会社内

Ｆターム(参考)　5B049 CC21 FF01 GG02
5E501 AC13 BA20 DA02 FA04 FA41